EvalEx

Software Requirements Specifications

Version 1.0

Revision History

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
| 08/10/23 | 1.0 | Added all the initial project software requirements specifications for EvalEx | Tommy Lam, Will Whitehead, Joshua Lee, Joe Hotze, |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

1.1 Purpose 4

1.2 Scope 4

1.3 Definitions, Acronyms, and Abbreviations 4

1.4 References 4

1.5 Overview 4

2. Overall Description 5

2.1 Product perspective 5

2.1.1 System Interfaces 5

2.1.2 User Interfaces 5

2.1.3 Hardware Interfaces 5

2.1.4 Software Interfaces 5

2.1.5 Communication Interfaces 5

2.1.6 Memory Constraints 5

2.1.7 Operations 5

2.2 Product functions 5

2.3 User characteristics 5

2.4 Constraints 5

2.5 Assumptions and dependencies 5

2.6 Requirements subsets 5

3. Specific Requirements 5

3.1 Functionality 5

3.1.1 <Functional Requirement One> 6

3.2 Use-Case Specifications 6

3.3 Supplementary Requirements 6

4. Classification of Functional Requirements 6

5. Appendices [6](#_qsh70q)

Software Requirements Specifications

# Introduction

## Purpose

The purpose of this SRS is to explain the nonfunctional requirements, design constraints, and other factors necessary for the project.

## Scope

This software will be a C++ program that can parse and evaluate arithmetic expressions. It will handle various arithmetic operators, numeric constants, and expressions with parentheses to define precedence and grouping.

## Definitions, Acronyms, and Abbreviations

PEMDAS: Stands for Parentheses, Exponents, Multiplication and Division, and Addition and Subtraction. It represents the order of operations in arithmetic.

## References

Team Happiness Club Software Development Plan - **[1]**

## Overview

The subsequent sections detail the overall description and specific requirements of the software project.

# Overall Description

## Product perspective

### System Interfaces

None, as this is a standalone software.

### User Interfaces

A user-friendly and legible command-line interface allowing users to input expressions and see calculated

results.

### Hardware Interfaces

No specific hardware interfaces are required, except standard input/output devices like keyboard and

monitor.

### Software Interfaces

Language: C++.

### Communication Interfaces

None required as this is a standalone application.

### Memory Constraints

Standard memory allocation for a basic C++ application.

### Operations

The program shall accept arithmetic expressions, evaluate them according to the PEMDAS rule, and

provide the result or a relevant error message.

## Product functions

* Parse and evaluate arithmetic expressions.
* Handle various operators (+, -, \*, /, %, ^).
* Handle numeric constants.
* Support parentheses for defining precedence and grouping.

## User characteristics

Users are expected to have basic knowledge of arithmetic expressions and the use of command-line

interfaces.

## Constraints

* Inputs should be valid arithmetic expressions.
* Expects integer inputs initially but may be extended to floating-point numbers in the future.
* The software should use object-oriented programming principles.

## Assumptions and dependencies

* The software will run on environments that support C++.
* Users are familiar with the PEMDAS rule.

## Requirements subsets

No subsets at this time.

# Specific Requirements

## Functionality

* Expression Parsing: Taking in user input into the program accounting operator precedence..
* Operator Support: Evaluating expressions supporting +,-,\*,/,%, operators..
* Parenthesis Handling: Evaluating expressions within parentheses first.
* Numeric Constants: Recognizing and calculating numeric constants.
* User Interface: Command-line interface for user interaction.
* Error Handling: Managing error scenarios, like division by zero or invalid expressions.

### <Functional Requirement One - Expression Parsing>

The program will be able to parse arithmetic expressions entered by the user, taking into account operator precedence and parenthesis.

*3.1.2 <Functional Requirement Two - Operator Support>*

The program will have operator support for addition, subtraction, multiplication, division, modulo, and exponentiation.

*3.1.3 <Functional Requirement Three - Parenthesis Handling>*

The program will be able to handle expressions enclosed within parentheses to determine the order of evaluation.

*3.1.4 <Functional Requirement Four - Numeric Constants>*

The program will be able to recognize and calculate numeric constants within the expression.

*3.1.5* *<Functional Requirement Five - Error Handling>*

The program will be able to manage error scenarios, like division by zero or invalid expressions.

*3.1.6* *<Functional Requirement Five - User Interface>*

The program will have a user-friendly, legible command-line interface. Users will be able to enter

expressions and the program will display the calculated result.

## Use-Case Specifications

Use cases include: Needing to do basic arithmetic of two numbers, in addition to following all pemdas rules. The user will be needing to perform arithmetic on numbers.

1. User starts the program.
2. User enters an arithmetic expression.
3. Program evaluates and displays the result or an error message.
4. User exits or continues with another expression.

## Supplementary Requirements

Non-functional requirements include:

* Program must execute in a timely manner.

Constraints include:

* Program must be written in C/C++.
* The software should provide clear documentation and comments.
* Include unit tests to verify the correctness of the expression evaluator.

# Classification of Functional Requirements

| **Functionality** | **Type** |
| --- | --- |
| Add operator support | Essential |
| Handle parentheses | Essential |
| Define operator according PEMDAS rules | Essential |
| Add user interface | Essential |
| Error handling | Essential |
| Accommodate for float value input | Desirable |

# Appendices

* Valid Expressions
  + (Refer to the project details for valid expression examples.)
* Invalid Expressions
  + (Refer to the project details for invalid expression examples.)